ON REDUCTION OF A MATRIX OF MUTUALLY EXCLUSIVE STATES AND ITS APPLICATION TO SIMULATION MODELS

by

William F. Drake

Neil J. Carscadden
Canadian Armed Forces

This paper contains a description of a techniquo developed to permit the efficient simulation of conflicting maintenance. It does not necessarily reflect Air Force Policy.

April 1968

MEMORANDUM REPORT NR. 3
Simulation Division
Advanced Logistics Systems Center
Headquarters Air Force Logistics Command

ABSTRACT

This paper presents the results of an effort to reduce the computer storage and execution time requirements for storing and processing matrices representing non-compatible system states or conflicts. Using previous methods the application of a conflicting matrix to simulation models was wasteful of time and/or storage. The method described provides the capability to apply a large matrix to simulation models with a greatly reduced requirement for core and processing time. Examples are given applying the method to conflicting maintenance on aircraft.

CONTENTS

	PAGE
INTRODUCTION	1
DEFINITIONS	2
OBJECTIVES	2
METHOD DESCRIPTION	3
EXAMPLE	5
LIMITATIONS	8
CORE REQUIREMENTS	8
ADDITIONAL REQUIREMENTS	9
SUMMARY	10

LIST OF FIGURES

FIGURE NO.	DESCRIPTION	PAGE
1	TASK MATRIX	2
2	CONFLICTING MATRIX	5

ON REDUCTION OF A MATRIX OF MUTUALLY EXCLUSIVE STATES AND ITS APPLICATION TO SIMULATION MODELS

INTRODUCTION

In many processes certain events will conflict with each other; i.e., they may have to be prevented from occurring simultaneously, or a third event may be necessary because of the conflict. This causes programming difficulties in many specific purpose models, and could require an enormous amount of core storage in general purpose models. This paper presents a unique method, which through the use of information in binary form, results in the ability to efficiently include and process a conflicting matrix in a model with substantial reductions in storage, programming, and execution time requirements.

This method was developed by the authors primarily for the Logistics Composite Model* and was used to represent conflicting maintenance on aircraft. The technique involved in this method is presented to assist those interested in its application to other simulation models.

*A. J. Clark, R. R. Fisher, W. F. Drake, and J. J. Delfausse, Logistics Composite Model (L-COM), The RAND Corporation, RM-5544-PR (DRAFT), November 1967.

DEFINITIONS

Given the task matrix shown in Figure 1, the following are defined:

1. Task - a particular item of work requiring time to accomplish, i.e., Tasks A, B, and C.

2. State - the combination of 0, 1, 2, ..., N (where N is dimension of the matrix) different tasks in process at the same time, i.e., "zero" A, B, C, AB, AC, BC or ABC, where AB = BA, etc.

3. Conflicts - states which by the matrix definition (X's) are forbidden, i.e., states AB, BC.

4. Exponent - the number of times a number is multiplied by itself, e.g., the binary exponent of 64 is 6.

5. Binary Power - the number "2" raised to an exponent, e.g., $1 = 2^0$, $2 = 2^1$, $4 = 2^2$, $8 = 2^3$, ...

Task	A	B	C
A		X	
B	X		X
C		X	

FIGURE 1 - TASK MATRIX

OBJECTIVES

Any procedure of this type must be able to:

1. provide a system identifier so that the current

system state can be determined quickly and efficiently;

2. store all defined conflicts; and

3. compare proposed states to the conflicts when necessary to see if the system is about to enter a conflicting or forbidden state.

METHOD DESCRIPTION

Each task defined in a conflicting matrix is assigned a binary power. The exponent is simply its row (or column) in the matrix, minus one; e.g., in Figure 1 the exponent of Task B is 1.

The system state identifier is a counter whose value is equal to the sum of the binary powers assigned to the tasks in process. Thus, when a system or procedure enters a new state, its counter is incremented by the binary power assigned to the new task; and when the system leaves the state, its counter is decremented by the same amount. A limitation of this method is that the state of the system cannot be made up of two or more identical tasks at the same time, i.e., a task must conflict with itself, an implied conflict.

Each conflict is maintained in computer memory by storing the indices of the matrix element which represents the conflicting state. Each index minus one is used as a

binary exponent, and the binary word formed using these exponents represents a state of the system in which the conflicting tasks would be processed simultaneously. These indices are stored in two lists so that the values in the corresponding list positions represent the conflict.*

The proposed state of the system is compared with the conflicts using a "logical AND" function. This function compares two binary words, bit by bit, and gives a product whose n^{th} bit is "1" only if the n^{th} bits of both the comparison words are "1". Thus, when the indices of the conflicts are used as exponents to make up a binary word, the result is a word with "1" bits in two positions. By using the AND** to compare this word "K" to the counter representing the next state of the system, the result will be equal to "K" only if the system counter contains the same two "1" bits, i.e., if the system would be in a state in which the two conflicting tasks were both being processed. Thus, by checking the new count before a task is started it can be determined if this would be a forbidden state.

---

* Other methods of storage would serve equally well, e.g., storing the two values in the same list with the conflict being represented by two successive values.

**AND (capitalized) refers to logical AND function.

EXAMPLE

Consider the matrix in Figure 2. This represents a system in which Task A conflicts with Tasks B and F; Task B conflicts with Tasks A, C, and E; etc. Thus, it can be seen that the system may be in state ACE, but state ACF is forbidden (Task A conflicts with Task F). It should be noted that in the example, as in all conflicting matrices, the shaded portion duplicates the non-shaded portion and can thus be eliminated from consideration. Also, as stated previously, a task must conflict with itself. These implied conflicts need not be stored in the matrix.

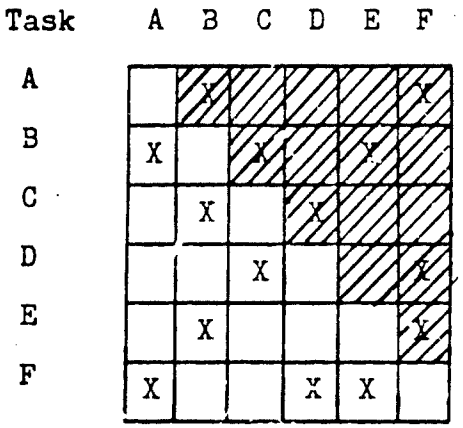

FIGURE 2 - CONFLICTING MATRIX

This information is stored in three lists - the first containing the names (or numbers) of the tasks in the matrix;

and the other two containing the conflicts (X's) in the form of the matrix indices. For this example, the lists (disregarding shaded area) would be:

TASKS	CONFLICTS	
LIST 1	LIST 2 (INDEX I)	LIST 3 (INDEX J)
A	2	1
B	3	2
C	4	3
D	5	2
E	6	1
F	6	4
	6	5

To illustrate the method of representing the states within a computer, a six-bit word will be used. Assume the system enters a state consisting of Task A. This task is in the first position in the list so the binary word representing the system counter will be 2^{1-1} or 2^{0}:

000001 = (counter)

Next, assume that Task B is to be started. This task is in the second position in the list so we will add 2^{2-1} to the system counter:

000011 = (counter)

Then a check is made to see if this state is a forbidden one. The first set of conflict indices are "2" and "1", which means the binary exponents are these values minus one. The word formed from these exponents is (2^1 plus 2^0):

000011 = (K)

We see that (counter) AND (K) = K; therefore, the system cannot enter this state. Task B cannot be started so the counter is reset to reflect the proper state (Task A).

As a second case, assume we are in state ACE:

010101

and want to start Task F, which would make the counter

110101

The indices of each of the seven conflicts give the following seven words:

000011 = (K1)

000110 = (K2)

001100 = (K3)

010010 = (K4)

100001 = (K5)

101000 = (K6)

110000 = (K7)

We can see that:

(counter) AND (K5) = K5

(counter) AND (K7) = K7

meaning this task is doubly forbidden (Task F conflicts with Tasks A and E) so the system cannot enter this state. Being only singly forbidden is sufficient, of course, to prevent the system from entering a state.

LIMITATIONS

This method is limited only secondarily by core storage availability, the primary limitation being the number of bits available in a computer word. The counter for an N x N matrix must be capable of storing a number as large as 2^N-1; so if a machine has a 36-bit word, then the limitation with a one-word counter is to a 36 x 36 matrix. However, by inserting more complicated coding, it is possible to circumvent this limit and use additional words to represent 2^{36} to $2^{71}-1$, 2^{72} to $2^{107}-1$, etc. The complexity of the additional coding goes up as the factorial of the number of words used for the counter.

A listing of the coding used in L-COM is contained in Appendix A. This application used two 36-bit words for the counter and so was able to process a 72 x 72 matrix of conflicting states.

CORE REQUIREMENTS

The primary consideration with respect to core storage requirement is the number of conflicts within the matrix, and the size of the matrix is of secondary importance.

The number of words required for an N x N matrix containing M conflicts is:

$$N + 2M$$

If partial word transfers (packing) were available, the requirement would be:

$$N \times \text{packing} + 2M \times \text{packing}$$

The computer available to the authors permitted packing of variables in 1/2, 1/3, 1/4 and 1/6 of a 36-bit word. In L-COM the matrix size was limited to 63 x 63 so that the largest indices (63) could be 1/6 packed. The task numbers required 1/3 packing so the total storage requirement for data was less than 200 words. The coding required to process this data in the model was less than 300 words of SLEUTH instructions.

ADDITIONAL REQUIREMENTS

The "AND" function is a necessity for this method regardless of the (simulation) language used. Most computers, however, would have it available in some manner. For example, it would be available as a FORTRAN V function or in an assembly language sub-routine for SIMSCRIPT* compilers;

*H. M. Markowitz, J. C. Hausner, and H. W. Karr, SIMSCRIPT: A Simulation Programming Language, The RAND Corporation, RM-3310-PR (DDC No. AD 291 806), November 1962.

and as an assembly language insert for SIMSCRIPT 1.5** compilers (e.g., SCL (selective clear) in SLEUTH, ANA in MAP).

SUMMARY

The application of this method to a simulation model gives it the capability to simulate a more realistic world, by explicitly considering conflicting states. This has generally been avoided or abstracted in other models. The examples given here have been concerned with conflicting maintenance, but the method is potentially applicable to other types of modeling problems.

**H. M. Markowitz, H. Kleine, H. W. Karr, SIMSCRIPT 1.5, California Analysis Center, Santa Monica, California.

APPENDIX A

CODING FROM L-COM

This listing is included only as a further example of the method and is not presumed to be entirely efficient or entirely correct.

```
C     THE FOLLOWING IS FROM SUBROUTINE START , WHICH CHECKS WHETHER
C          RESOURCES ARE AVAILABLE TO START A TASK. FIRST A CHECK IS
C          MADE TO SEE IF THE TASK CONFLICTS.(WRITTEN IN SIMSCRIPT  1.5)
C               IC = TASK NAME (ID NUMBER)
C             NTCON= NUMBER OF CONFLICT TASKS = DIMENSION OF MATRIX
C          TKIND(I)=NAME OF TASK IN ITH POSITION OF MATRIX
C             NEXES= NUMBER OF X,S IN CONFLICT MATRIX
C          CONID(K) AND CNID2(K) ARE LISTS CONTAINING INDICES OF
C                   CONFLICTS (X'S)
      LET NW1=1
      LET ICK=0
C     IF TDSTP(IC) IS LESS THAN ZERO, THEN TASK IC IS IN CONFLICT MATRIX
      IF(TDSTP(IC))GE(0),GO TO 88
      FIND FIRST, FOR I=(1)(NTCON),WITH(TKIND(I))EQ(IC),WHERE KTK,
     2  IF NONE, GO TO ERR
C     NET IS AN ENTITY WHICH IS ABLE TO KEEP SYSTEM COUNTERS
      LET KCT=KNT1(NET)
      LET KCT1=KNT2(NET)
      IF (KTK)LE(35),GO TO B1
C       BY SUBTRACTING 36 FROM AN INDEX TO OBTAIN WORD NC1,
C          THE SECOND WORD OF A TWO-WORD COUNTER CAN BE USED
      LET NC=0
      LET NC1=2**(KTK-36)
      LET NW1 = 3
      GO TO B2
  ERR WRITE ON 6,IC
      FT(*1 TDSTP(*,I4,*) NOT LE ZERO BUT TKIND NOT FOUND*)
      RETURN
   B1 LET NC1=0
      LET NC =2**(KTK-1)
      LET NW1=2
   B2 CALL STCON(ICK,KCT,NC,KCT1,NC1,NW1)
C     TASK CONFLICTS WITH SELF IF ICK=1
      IF(ICK)NE(1),GO TO B4
C     IF JJ GREATER THAN ONE THEN TASK HAS BEEN BACKORDERED PREVIOUSLY
C                                          JJ=2  - FOR RESOURCES
C                                          JJ=3  - FOR A CONFLICT
   B3 IF(JJ)GE(2),RETURN
      LET ITT =NET
      GO TO 525
C   525 IS SECTION OF CODING WHICH BACKORDERS THE TASK
```

```
   B4 LET KCT=KCT+NC
      LET KCT1=KCT1+NC1
      DO TO B5,FOR I=(1)(NEXES)
      LET K =CONID(I)
      LET M  =CNID2(I)
       IF(K)GR(35),GO TO D1
      LET NC=2**(K-1)
      LET NC1=0
      GO TO D2
   D1 LET NC1=2**(K-36)
      LET NC=0
   D2 IF(M)GR(35),GO TO D3
      LET NC=NC+2**(M-1)
      GO TO D4
   D3 LET NC1=NC1+2**(M-36)
   D4 CALL STCON(ICK,KCT,NC,KCT1,NC1,0)
      IF(ICK)EQ(1),GO TO B3

   B5 LOOP
C      ICK WAS NEVER EQUAL TO 1, THEREFORE THIS TASK DOESN'T CONFLICT
      IF(JJ)NE(3),GO TO B8
C     SUBROUTINE PULL TAKES THE TASK OUT OF BACKORDER
      CALL PULL(IJ,NET)
   B8
C     *     *     *     *     *     *     *     *
C     THIS SECTION WOULD DETERMINE  WHETHER TASK HAS RESOURCES TO START
C     *     *     *     *     *     *     *     *
C     IF THIS SECTION OF CODING IS REACHED, THEN TASK CAN START
C     IF TASK IS IN CONFLICT MATRIX, UPDATE SYSTEM COUNTER
      IF(NW1)EQ(1),GO TO C5
      LET KNT1(NET)=KCT
      LET KNT2(NET) =KCT1
C      JBEND IS EVENT NOTICE OF TASK BEING STARTED
   C5 CAUSE JBEND CALLED IJ AT DL
      RETURN
      END

      SUBROUTINE STCON(NGC,KCT,NC,KCT1,NC1,NW1)
C      THIS SUBROUTINE IS WRITTEN IN FORTRAN V
      INTEGER AND
      IF(AND(KCT ,NC ).EQ.NC ) GO TO 5
      IF(NC.NE.0) GO TO 9
      GO TO 8
    5 IF(NW1.EQ.2) GO TO 12
    8 IF(AND(KCT1,NC1).EQ.NC1) GO TO 12
      IF(NC1.EQ.0) GO TO 12
    9 NGC =0
      GO TO 13
   12 NGC =1
   13 RETURN
      END
```


12

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) Headquarters Air Force Logistics Command Wright-Patterson AFB, Ohio	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3 REPORT TITLE

On Reduction of a Matrix of Mutually Exclusive States and Its Application to Simulation Models

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

5 AUTHOR(S) (First name, middle initial, last name)

William F. Drake and Neil J. Carscadden (Captain) Canadian Armed Forces

6 REPORT DATE April 1968	7a. TOTAL NO. OF PAGES 17	7b. NO. OF REFS 3
8a. CONTRACT OR GRANT NO Independent Research b. PROJECT NO c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ACLS MR 3 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10 DISTRIBUTION STATEMENT

Distribution of this Document is unlimited.

11 SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13 ABSTRACT

This paper presents the results of an effort to reduce the computer storage and execution time requirements for storing and processing matrices representing non-compatible system states or conflicts. Using previous methods the application of a conflicting matrix to simulation models was wasteful of time and/or storage. The method described provides the capability to apply a large matrix to simulation models with a greatly reduced requirement for core and processing time. Examples are given applying the method to conflicting maintenance on aircraft.

DD FORM 1 NOV 65 1473

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Simulation						
Conflicting Maintenance						
Conflict Matrix						
Mutually Exclusive States						
Matrix Reduction						